Academic Year: 2025-26

LABORATORY MANUAL

Name of the Student:		
Class: BE	**Division: B**	**Roll No.:**
Subject: DMV (2019 Course) [417522]		**Exam Seat No.:**
Department of Artificial Intelligence and Data Science		

Program Outcomes (PO's):

POs are statements that describe what students are expected to know and be able to do upon graduating from the program. These relate to the skills, knowledge, analytical ability attitude and behavior that students acquire through the program.

- **PO1: Engineering Knowledge:**
  Graduates will be able to apply the Knowledge of the mathematics, science and engineering fundamentals for the solution of engineering problems related to IT.

- **PO2: Problem Analysis:**
  Graduates will be able to carry out identification and formulation of the problem statement by requirement engineering and literature survey.

- **PO3: Design/Development of Solutions:**
  Graduates will be able to design a system, its components and/or processes to meet the required needs with consideration for public safety and social considerations.

- **PO4: Conduct Investigations of Complex Problems:**
  Graduates will be able to investigate the problems, categorize the problem according to their complexity using modern computational concepts and tools.

- **PO5: Modern Tool Usage:**
  Graduates will be able to use the techniques, skills, modern IT engineering tools necessary forengineering practice.

- **PO6: The Engineer and Society:**
  Graduates will be able to apply reasoning and knowledge to assess global and societal issues

- **PO7: Environment and Sustainability:**
  Graduates will be able to recognize the implications of engineering IT solution with respect to society and environment.

PO8: Ethics:

Graduates will be able to understand the professional and ethical responsibility.

- **PO9: Individual and Team Work:**

  Graduates will be able to function effectively as an individual member, team member or leader in multi -disciplinary teams.

- **PO10: Communication:**

  Graduates will be able to communicate effectively and make effective documentations and presentations.

- **PO11: Project Management and Finance:**

  Graduates will be able to apply and demonstrate engineering and management principles in project management as a member or leader.

- **PO12: Life-long Learning:**

  Graduates will be able to recognize the need for continuous learning and to engage in life-long learning.

Course Objectives and Course Outcomes(COs)

Course Objectives:

- Apply regression, classification and clustering algorithms for creation of ML models
- Introduce and integrate models in the form of advanced ensembles.
- Conceptualized representation of Data objects.
- Create associations between different data objects, and the rules.
- Organized data description, data semantics, and consistency constraints of data

Course Outcomes:

On completion of the course, students will be able to–

CO1: Implement regression, classification and clustering models

CO2: Integrate multiple machine learning algorithms in the form of ensemble learning.

CO3: Apply reinforcement learning and its algorithms for real world applications.

CO4: Analyze the characteristics, requirements of data and select an appropriate data model.

CO5: Apply data analysis and visualization techniques in the field of exploratory data science

CO6: Evaluate time series data.

Dr. D. Y. Patil Educational Federation's

Dr. D. Y. PATIL COLLEGE OF ENGINEERING &INNOVATION

Department of Artificial Intelligence and Data Science

Academic Year 2025-26

CERTIFICATE

This is to certify that Mr. /Ms. _____

of Class BE - AI-DS, Roll No. Examination Seat No. _____

has completed all the practical work in the Computer Laboratory - I [417522] satisfactorily, as prescribed by Savitribai Phule Pune University, Pune in the academic year 2025-26 (Term-I).

Place:

Date:

Course In-charge **Department of** **AI-DS**	**HOD** **Department of** **AI-DS**	**Principal** **DYPCOEI,** **Varale**

INDEX

CL-I (Data Modeling and Visualization Laboratory) **Class:** B.E.

Sr. No.	Name of the Experiment	Date of Conduction	Date of Checking	Page No.	Sign	Remark
1	**Data Loading, Storage and File Formats:** Analyzing Sales Data from Multiple File Formats Dataset: Sales data in multiple fileformats (e.g.,CSV,Excel, JSON) Description: The goal is to loadand analyzesales data from different file formats,including CSV,Excel, and JSON, and perform data cleaning, transformation, and analysis on the dataset.					
2	**Interacting with Web APIs Problem Statement:** Analyzing Weather Data from Open Weather Map API Dataset: Weather data retrieved from Open Weather Map API Description: The goal is to interact with the Open Weather Map API to retrieve weather data for a specific location and perform data modeling and visualization to analyze weather patterns over time.					

3	**Data Cleaning and Preparation Problem Statement:** Analyzing Customer Churn in a Telecommunications Company Dataset: "Telecom_Customer_Churn.csv" Description: The dataset Contains information about customers of a telecommunications company and whether they have churned (i.e., discontinued their services). The dataset includes various attributes of the customers, such as their demographics, usage patterns, and account information. The goal is to perform data cleaning and preparation to gain insights into the factors that contribute to customer churn.					
4	**Data Wrangling Problem Statement:** Data Wrangling on Real Estate Market Dataset: "RealEstate_Prices.csv" Description: The dataset contains information about housing prices in a specific real estate market. It includes various attributes such as property characteristics, location, sale prices, and other relevant features. The goal is to perform data wrangling to gain insights into the factors influencing housing prices and prepare the dataset for further analysis or modeling.					

5	**Data Visualization using matplotlib Problem Statement:** Analyzing Air Quality Index (AQI) Trends in a City Dataset: "City_Air_Quality.csv" Description: The dataset contains information about air quality measurements in a specific city over a periodof time. It includes attributes such as date, time, pollutant levels (e.g., PM2.5, PM10, CO), and the Air Quality Index (AQI) values.The goal is to use the matplotlib library to create visualizations that effectively representthe AQI trends and patterns for different pollutants in the city.					
6	**Time Series Data Analysis Problem** statement: Analysis and Visualization of Stock Market Data Dataset: "Stock_Prices.csv" Description: The dataset contains historical stock price data for a particular company over a period of time. It includes attributes such as date, closing price, volume, and other relevant features. The goal is to perform time series data analysis on the stock price data to identify trends, patterns, and potential predictors, as well as build models to forecast future stock prices.					

Experiment No: 1

Analyzing Sales Data from Multiple File Formats

Name of the Student: _____

Class: BE **Batch:**

Date:	Mark: /10

Signature of the Course In-charge: _____

Signature of the HOD: _____

EXPERIMENT NO.1

Practical Title: Study Analyzing Sales Data from Multiple File Formats

Aim: To load, clean, transform, and analyze sales data from various file formats (CSV, Excel, JSON)and to derive meaningful insights through data visualization.

Objective:

- To Import sales data from CSV, Excel, and JSON file formats into appropriate data structures.
- To Identify and address data quality issues such as missing values or inconsistencies.To
- Perform data cleaning operations to standardize the dataset.
- To Convert the data into a unified format and perform necessary transformations.To
- Conduct descriptive analysis to understand sales metrics and trends.
- To Create visualizations to represent and interpret the sales data.

Software Requirements:

- Anaconda with Python 3.7
- Libraries: Pandas, NumPy, Matplotlib, Seaborn (install using pip)
- Jupyter Notebook (optional, for an interactive environment)

Hardware Requirement:

- PIV, 2GB RAM, 500 GB HDD.

Learning Objectives:

To Learn Analyzing Sales Data from Multiple File Formats.

Outcome:

1. **Cleaned and Unified Dataset:** A single, cleaned data frame containing the merged data fromCSV, Excel, and JSON files.
2. **Descriptive Analysis Results:** Metrics such as total sales, average order value, and distribution byproduct category.
3. **Visualizations:** Graphical representations of sales trends, customer behavior, and productperformance.

Theory:

Data Loading

- **CSV Files**: CSV (Comma-Separated Values) files are text files where each line represents a data record, and each record consists of fields separated by commas. The pandas.read_csv() function inPython efficiently loads this data into a DataFrame.
- **Excel Files**: Excel files are spreadsheet files that can store data in multiple sheets. The pandas.read_excel() function allows loading data from specific sheets or all sheets into DataFrames.

- **JSON Files**: JSON (JavaScript Object Notation) files store data in a hierarchical format that can represent complex data structures. The pandas .read json() functionloads this data into a DataFrameby converting the nested JSON objects into tabularformat.

Data Cleaning

- **Handling Missing Values**: Missing values can be addressed by either imputing them (using mean,median, or mode) or removing the rows/columns containing them, depending on the context.
- **Removing Duplicates**: Duplicate entries can skew analysis. The drop_duplicates() method in pandashelps in identifying and removing these duplicates.
- **Correcting Inconsistencies**: This includes standardizing data formats (e.g., date formats), correctingtypos, and ensuring consistent categorization.

Data Transformation

- **Merging Datasets**: Combining data from different sources (e.g., CSV, Excel, JSON) into a unifiedDataFrame is crucial for a comprehensive analysis. Functions like merge() or concat() in pandas facilitate this.
- **Feature Engineering**: Creating new features from existing data can enhance the analysis. Forinstance, creating a "total revenue" column by multiplying quantity and price.
- **Normalization**: Standardizing the range of data values to improve the performance of somealgorithms. Techniques such as min-max scaling or z-score normalization are used.

Data Analysis

- **Descriptive Statistics**: Basic metrics like mean, median, mode, standard deviation, and percentilesprovide insights into the central tendency and dispersion of the data.
- **Aggregation**: Summarizing data at different levels, such as total sales per region or average ordervalue per customer, helps in understanding broader patterns.

Data Visualization

- **Bar Plots**: Useful for comparing quantities across different categories. For example, comparing totalsales by product category.
- **Line Charts**: Effective for showing trends over time, such as monthly sales growth.
- **Pie Charts**: Useful for depicting the percentage distribution of categories, such as sales distributionby region.
- **Box Plots**: Useful for showing data distribution and identifying outliers.

Algorithms

1. **Outlier Detection**:
   - **Z-Score**: Measures how far away a data point is from the mean, in terms of standarddeviations. Points with a z-score beyond a certain threshold are considered outliers.
   - **IQR (Interquartile Range)**: Outliers are identified by checking if data points fall below Q1 -1.5*IQR or above Q3 + 1.5*IQR, where Q1 and Q3 are the first and third quartiles, respectively.
2. **Clustering (for deeper insights)**:
   - **K-Means Clustering**: Groups data points into clusters based on their similarity. Useful forcustomer segmentation by purchasing behavior.
   - **Hierarchical Clustering**: Builds a hierarchy of clusters, useful for understanding the data'sstructure at different levels of granularity.
3. **Regression Analysis (for trend analysis)**:
   - **Linear Regression**: Analyzes the relationship between two variables, such as sales andadvertising spend, to predict future sales.

Conclusion:

The practical successfully demonstrated the process of loading, cleaning, transforming, and analyzing sales data from various file formats. By consolidating the data into a unified format, performing descriptive analysis, and visualizing the results, we gained valuable insights into sales trends and product performance.This process ensures that data-driven decisions can be

made with a high degree of accuracy and confidence.

Viva Questions:

1. What is sales data, and why is it important for businesses?
2. What were the objectives of your sales data analysis project?
3. What types of insights did you hope to derive from the analysis of the sales data?

Coding Efficiency	Viva	Timely Completion	Total	Dated Sign of Course In-charge
5	3	2	10	

Program:

Experiment No: 2

Analyzing Weather Data from Open Weather Map API

Name of the Student: _____

Class: BE **Batch:**

Date:	Mark: /10

Signature of the Course In-charge: _____

Signature of the HOD: _____

EXPERIMENT NO.2

Practical Title: Analyzing Weather Data from Open Weather Map API

Aim:

To interact with the Open Weather Map API to retrieve and analyze weather data for a specific location, and to model and visualize weather patterns over time.

Objectives:

- **To API Registration:** Obtain an API key from Open Weather Map for accessing weather data.
- **To Data Retrieval:** Use the API key to fetch weather data for a specific location.
- **To Data Extraction:** Extract relevant weather attributes such as temperature, humidity, wind speed, and precipitation from the API response.
- **To Data Cleaning:** Preprocess and clean the data, addressing missing values and inconsistencies.
- **To Data Modeling:** Analyze weather patterns by calculating average temperatures, extreme values, and trends.
- **To Data Visualization:** Create visualizations to represent temperature changes, precipitation levels, and wind speed variations.
- **To Data Aggregation:** Summarize weather statistics by specific time periods such as daily, monthly, or seasonal.
- **To Geospatial Visualization:** Incorporate geographical information to visualize weather patterns across different locations.
- **To Correlation Analysis:** Explore relationships between weather attributes using correlation plots or heat maps.

Software and Hardware Requirements:

- **Software:**
  - Python 3.x
  - Libraries: requests, pandas, matplotlib, seaborn, geopandas (install using pip)
  - Jupyter Notebook (optional, for interactive analysis)
- **Hardware:**
  - A computer with at least 4 GB of RAM
  - Internet connection for API requests
  - Enough storage to handle data files and visualizations

Expected Outcome:

1. **Cleaned Weather Data:** A dataset containing weather attributes for analysis.
2. **Descriptive Statistics:** Metrics such as average temperature, maximum/minimum values, and weather trends.
3. **Visualizations:** Graphical representations of weather data trends, precipitation levels, and wind speed variations.
4. **Geospatial Maps:** Maps showing weather patterns across different locations (if geographical data is available).
5. **Correlation Insights:** Heatmaps or plots showing relationships between weather attributes.

Theory:

API Interaction

- **Overview**: The OpenWeatherMap API provides weather data in JSON format. To access this data, you need to send HTTP requests with your API key.
- **Steps**:
    1. **Obtain API Key**: Register on the OpenWeatherMap platform to get an API key.
    2. **Send HTTP Requests**: Use libraries such as requests in Python to send GET requests to the API endpoint.
    3. **Receive JSON Response**: The API responds with weather data in JSON format.
    4. **Parse JSON Data**: Use libraries like json in Python to parse the JSON response and extract relevant information.

Data Cleaning

- **Overview**: Ensures the dataset is accurate and ready for analysis.
- **Steps**:
    1. **Handle Missing Values**: Impute or remove missing values.
    2. **Ensure Data Consistency**: Verify that all data entries conform to expected formats and ranges.
    3. **Convert Data Types**: Convert data types to the appropriate format, e.g., timestamps to datetime objects.

Data Modeling

- **Overview**: Involves statistical analysis to derive insights from the data.
- **Steps**:

1. **Calculate Statistical Metrics**: Compute metrics such as mean, median, and standard deviation.
2. **Identify Trends**: Use time-series analysis to observe patterns and trends over time.
3. **Model Forecasts**: Apply predictive models (e.g., ARIMA) to forecast future weather conditions.

Data Visualization

- **Overview**: Techniques for visually representing data to uncover insights.
- **Steps**:
  1. **Line Charts**: Display changes in weather variables over time.
  2. **Bar Plots**: Compare different weather attributes or locations.
  3. **Scatter Plots**: Show relationships between two weather variables.

Geospatial Visualization

- **Overview**: Uses geographical data to illustrate spatial patterns.
- **Steps**:
  1. **Obtain Geospatial Data**: Get location coordinates and weather data.
  2. **Create Maps**: Use libraries like folium or geopandas to create visual maps representing weather conditions across different locations.
  3. **Overlay Weather Data**: Add weather data layers to the maps for spatial analysis.

Correlation Analysis

- **Overview**: Examines the relationships between different weather attributes.
- **Steps**:
  1. **Compute Correlation Coefficients**: Use Pearson or Spearman correlation to quantify relationships.
  2. **Visualize Correlations**: Use heatmaps or scatter plots to represent correlation between variables like temperature and humidity.

Algorithms

1. **Time-Series Analysis**:
   - **Moving Average**: Smooths out short-term fluctuations and highlights longer-term trends.

```
df['Moving_Avg'] = df['Temperature'].rolling(window=7).mean()
```

- **ARIMA (AutoRegressive Integrated Moving Average)**: Used for forecasting future values based on past data.

```
from statsmodels.tsa.arima_model import ARIMA
model = ARIMA(df['Temperature'], order=(5,1,0))
model_fit = model.fit(disp=0)
forecast = model_fit.forecast(steps=30)[0]
```

2. **Correlation Analysis**:
   - **Pearson Correlation Coefficient**: Measures linear correlation between two variables.

```
correlation = df['Temperature'].corr(df['Humidity'])
```

   - **Spearman Rank Correlation**: Measures the rank-order correlation.

```
from scipy.stats import spearmanr
correlation, _ = spearmanr(df['Temperature'], df['Humidity'])
```

3. **Geospatial Analysis**:
   - **K-Means Clustering**: Groups data points into clusters based on geographical location and weather attributes.

```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=5)

df['Cluster'] = kmeans.fit_predict(df[['Latitude', 'Longitude']])
```

   - **Heatmaps**: Visualize density of weather conditions across different locations.

```
import folium
heatmap = folium.Map(location=[latitude, longitude], zoom_start=12)
```

This extended theory provides a comprehensive approach to analyzing weather data, incorporating API interaction, data cleaning, modeling, visualization, and correlation analysis,along with a flow diagram and relevant algorithms.

Conclusion:

The practical demonstrated how to interact with the Open Weather Map API to retrieve and analyze weather data. By extracting and cleaning weather attributes, we were able to visualize and interpret the current weather conditions. For a comprehensive analysis, historical data and geospatial information would be needed to explore trends, aggregate statistics, and visualize weather patterns across different locations. This approach provides valuable insights into weather behavior and facilitates better decision-making based on weather data.

Viva Questions:

1. What is the Open Weather Map API, and how does it work?
2. What kind of data can you retrieve from the Open Weather Map API?
3. What insights did you gain from visualizing the weather data?

Coding Efficiency	Viva	Timely Completion	Total	Dated Sign of Course In-charge
5	3	2	10	

Program:

Experiment No: 3

Data Cleaning and Preparation

Name of the Student: _____

Class: BE **Batch:**

Date:

Mark: /10

Signature of the Course In-charge: _____

Signature of the HOD: _____

EXPERIMENT NO.3

Practical Title: Data Wrangling on Real Estate Market

Aim:

To perform data wrangling on a real estate market dataset to clean, preprocess, and prepare the data for analysis or modeling, with a focus on understanding factors influencing housing prices.

Objectives:

- **To Import Dataset:** Load the "RealEstate_Prices.csv" dataset and clean column names.
- **To Handle Missing Values:** Address missing values using appropriate strategies.
- **To Data Merging:** Merge with additional datasets if available to enrich the dataset.
- **To Filter and Subset Data:** Extract relevant data based on specific criteria such as time period, property type, or location.
- **To Encode Categorical Variables:** Transform categorical variables for analysis.
- **To Aggregate Data:** Compute summary statistics or derived metrics, such as average sale prices by neighborhood or property type.
- **To Identify and Handle Outliers:** Detect and manage outliers or extreme values that may affect the analysis.

Software and Hardware Requirements:

- **Software:**
  - Python 3.x
  - Libraries: pandas, numpy, matplotlib, seaborn (install using pip)
  - Jupyter Notebook (optional, for an interactive environment)
- **Hardware:**
  - A computer with at least 4 GB of RAM
  - Enough storage to handle data files and intermediate results

Expected Outcome:

- **Cleaned Dataset:** A well-prepared dataset with clean column names, handled missing values, and appropriate data types.
- **Merged Data:** Enhanced dataset with additional information from other relevant datasets.
- **Filtered Data:** Subset of data based on criteria like time period, property type, or location.
- **Encoded Variables:** Categorical variables encoded for analysis.
- **Aggregated Metrics:** Summary statistics, such as average sale prices by neighborhood or property type.
- **Outlier Management:** Identified and managed outliers to improve data quality.

Theory:

Data Wrangling is the process of cleaning, structuring, and enriching raw data into a suitable format for analysis. It involves several steps to ensure data quality and usability:

1. Importing Data

- **Overview**: Loading data from various sources into a suitable format for analysis.
- **Steps**:
  - **Read from File**: Use functions like pandas.read_csv(), pandas.read_excel(), or pandas.read_json() to load data.
  - **Address Formatting Issues**: Ensure correct data types, date formats, and encoding.

2. Handling Missing Values

- **Overview**: Addressing gaps in the dataset to maintain data integrity.
- **Techniques**:
  - **Imputation**: Fill missing values using statistical methods (mean, median, mode) or more complex algorithms (KNN imputation).
  - **Removal**: Delete rows or columns with missing values if they are insignificant or if imputation is not suitable.

3. Merging Datasets

- **Overview**: Combining multiple datasets to provide a comprehensive view.
- **Steps**:
  - **Concatenation**: Stack datasets vertically or horizontally using pandas.concat().
  - **Join/Merge**: Combine datasets based on a common key using pandas.merge() or pandas.join().

4. Filtering Data

- **Overview**: Selecting specific subsets of data based on criteria.
- **Steps**:
  - **Apply Conditions**: Use boolean indexing or pandas. query() to filter rows that meet certain conditions.

5. Encoding Categorical Variables

- **Overview**: Converting non-numeric data into numeric formats for analysis.
- **Techniques**:
  - **Label Encoding**: Convert categories to numeric labels.
  - **One-Hot Encoding**: Create binary columns for each category using pandas. Get _dummies().

6. Aggregating Data

- **Overview**: Summarizing data to derive meaningful metrics.
- **Steps**:

- **Group By**: Use pandas. Group by() to aggregate data by one or more columns.
- **Aggregation Functions**: Apply functions like sum(), mean(), count() to summarize data.

7. Handling Outliers

- **Overview**: Managing extreme values that may skew analysis.
- **Techniques**:
  - **Statistical Methods**: Identify outliers using z-scores or IQR (Interquartile Range).
  - **Visual Methods**: Use box plots or scatter plots to detect outliers visually.

Algorithms:-

1. **Imputation Techniques**
   - **Mean Imputation**: Fill missing values with the mean of the column.

     ```
     df['column_name'].fillna(df['column_name'].mean(), inplace=True)
     ```

   - **KNN Imputation**: Use K-Nearest Neighbors to impute missing values.

     ```
     from sklearn.impute import KNNImputer
     imputer = KNNImputer(n_neighbors=5)
     df_imputed = imputer.fit_transform(df)
     ```

2. **Outlier Detection**
   - **Z-Score Method**: Identify outliers based on Z-score.

     ```
     from scipy import stats
     df['z_score'] = stats.zscore(df['column_name'])
     outliers = df[df['z_score'].abs() > 3]
     ```

   - **IQR Method**: Use Interquartile Range to find outliers.

     ```
     Q1 = df['column_name'].quantile(0.25)
     Q3 = df['column_name'].quantile(0.75)
     IQR = Q3 - Q1
     outliers = df[(df['column_name'] < (Q1 - 1.5 * IQR)) | (df['column_name'] > (Q3 + 1.5 * IQR))]
     ```

3. **Encoding Categorical Variables**
   - **Label Encoding**: Convert categories to numerical values.

     ```
     from sklearn.preprocessing import LabelEncoder
     le = LabelEncoder()
     df['encoded_column'] = le.fit_transform(df['categorical_column'])
     ```

   - **One-Hot Encoding**: Create binary columns for categories.

```
df_encoded = pd.get_dummies(df, columns=['categorical_column'])
```

4. **Aggregating Data**
   - **Group By and Aggregation**:

```
df_grouped = df.groupby('category_column').agg({'value_column': 'sum'})
```

This extended theory provides a detailed approach to data wrangling, including a flow diagram and algorithmsfor each step, ensuring a thorough and systematic process for preparing data for analysis.

Conclusion:

The practical demonstrated the process of data wrangling on a real estate market dataset. By cleaning and preprocessing the data, merging additional datasets, and handling missing values, we prepared the dataset for further analysis. Filtering, encoding, and aggregating the data allowed us to derive insights into factors influencing housing prices. Managing outliers helped ensure the accuracy of subsequent analyses. This comprehensive data wrangling approach provides a solid foundation for detailed analysis and modeling in the real estate domain.

Viva Questions:

1. What is data wrangling, and why is it crucial for analyzing the real estate market?
2. What key attributes are typically included in real estate datasets?
3. What are the common data sources used in real estate market analysis?

Coding Efficiency	Viva	Timely Completion	Total	Dated Sign of Course In-charge
5	3	2	10	

Program:

Experiment No: 4

Data Wrangling

Name of the Student: _____

Class: BE **Batch:**

Date:

Mark: /10

Signature of the Course In-charge: _____

Signature of the HOD: _____

EXPERIMENT NO.4

Practical Title: Data Wrangling on Real Estate Market

Aim:

The aim of this project is to extract, clean, and process real estate market data to provide meaningful insights regarding property values, trends, and other market dynamics. This will involve handling messy, incomplete, or inaccurate data and transforming it into a structured format for analysis and decision-making.

Objectives:

1. **Data Collection**: Gather real estate market data from various sources, including property listings, government data, and third-party datasets (e.g., Zillow, Realtor).
2. **Data Cleaning**: Handle missing values, duplicate entries, and outliers in the dataset.
3. **Data Transformation**: Convert data into a consistent format, such as standardizing property sizes, prices, and locations.
4. **Data Enrichment**: Add external data (e.g., socio-economic or neighborhood data) to improve analysis.
5. **Exploratory Data Analysis (EDA)**: Explore and visualize data to discover trends, patterns, and correlations.
6. **Feature Engineering**: Create new features, such as price per square foot or neighborhood popularity, for deeper insights.
7. **Reporting**: Provide a summary of findings and insights, including trends in property prices, demand in different neighborhoods, and predictions for future market trends.

Software and Hardware Requirements:

Software:

1. **Python**: Programming language for data wrangling and analysis.
    - Libraries: Pandas, NumPy, Matplotlib, Seaborn, Scikit-learn (optional for advanced analysis)
2. **Jupyter Notebook or PyCharm**: For writing and running Python code.
3. **Google Colab (optional)**: Cloud-based Jupyter environment for handling larger datasets.
4. **SQL**: For querying databases (if the data is stored in relational databases).
5. **Tableau/Power BI**: For advanced visualization and reporting.

Hardware:

1. **Processor**: Intel Core i5 (or equivalent) or higher.
2. **RAM**: Minimum 8 GB RAM (16 GB recommended for large datasets).
3. **Storage**: SSD with at least 100 GB free space (for handling large datasets).
4. **Internet Connection**: For downloading datasets and using cloud services (if applicable).

Expected Outcome:

1. **Clean and Structured Data**: A cleaned and well-organized dataset ready for analysis.
2. **Trends and Insights**: Identified key trends, such as price fluctuations in different neighborhoods, popular property types, and factors affecting real estate prices.
3. **Visualizations**: Graphs, charts, and maps showing trends, relationships, and distributions in the real estate market.
4. **Report**: A comprehensive report that summarizes findings, offers insights into the real estate market, and provides recommendations or predictions.

Theory:

Data Wrangling involves transforming raw data into a useful format for analysis. In the context of the real estate market, raw data may come from multiple sources (websites, public records, APIs) and can be messy or incomplete. The process includes:

- **Cleaning**: Removing or correcting inaccurate data.
- **Normalizing**: Standardizing formats (e.g., currency, date formats).
- **Imputing**: Handling missing values using strategies such as mean imputation or interpolation.
- **Outlier Detection**: Identifying and handling anomalies in the dataset.
- **Merging**: Combining datasets from different sources (e.g., property prices with neighborhood data).

Algorithm (Simplified Workflow):

1. **Data Collection**:
   - Import datasets from various sources (e.g., CSV, JSON, SQL).
   - Example: pd.read_csv('real_estate_data.csv')
2. **Data Cleaning**:
   - Remove duplicates: df.drop_duplicates()
   - Handle missing values: df.fillna(method='ffill') or df.dropna()
   - Convert data types (e.g., string to datetime or numeric).
3. **Data Transformation**:
   - Normalize data (e.g., prices in different currencies): df['price'] = df['price'].apply(lambda x: convert_currency(x))
   - Standardize property sizes: df['size'] = df['size'].apply(lambda x: standardize_size(x))
4. **Data Enrichment**:
   - Merge additional data (e.g., crime rates, schools): df = df.merge(additional_data, on='location')

1. **Exploratory Data Analysis (EDA)**:
   - Visualize price trends: sns.lineplot(data=df, x='date', y='price')
   - Create histograms for property size: df['size'].hist()
2. **Feature Engineering**:
   - Calculate price per square foot: df['price_per_sqft'] = df['price'] / df['size']
3. **Reporting**:
   - Export data to Excel: df.to_excel('cleaned_real_estate_data.xlsx')
   - Generate reports and visualizations using Tableau or Power BI.

Conclusion:

Data wrangling is essential for transforming raw, unstructured real estate market data into a clean, usable format for analysis. By performing steps like data cleaning, transformation, and enrichment, we gain insights into market trends and improve decision-making. This project helps understand property price dynamics and make predictions for future developments in the real estate market.

Viva Questions:

1. What are the key steps involved in the data wrangling process?
2. What types of raw data are typically used in the real estate market for analysis?
3. What challenges did you face during the data wrangling process?

Coding Efficiency	Viva	Timely Completion	Total	Dated Sign of Course In-charge
5	3	2	10	

Program:

Experiment No: 5

Data Visualization using matplotlib

Name of the Student: _____

Class: BE **Batch:**

Date:	Mark: /10

Signature of the Course In-charge: _____

Signature of the HOD: _____

EXPERIMENT NO.5

Practical Title: Data Visualization of Air Quality Index (AQI) Trends Using Matplotlib

Aim:

To analyze and visualize the trends and patterns of air quality measurements in a city over time using the Matplotlib library, focusing on AQI and pollutant levels.

Objectives:

- **To Import Dataset:** Load the "City_Air_Quality.csv" dataset into a Data Frame.
- **To Explore Data:** Understand the dataset's structure, content, and relevant attributes.
- **To Identify Variables:** Determine which variables are pertinent for visualizing AQI and pollutant trends.
- **To Overall AQI Trend:** Create line plots to visualize the overall AQI trend over time.
- **To Pollutant Trends:** Plot individual pollutant levels (e.g., PM2.5, PM10, CO) to observe their trends.
- **To AQI Comparison:** Use bar plots or stacked bar plots to compare AQI values across different dates or time periods.
- **To Distribution Analysis:** Create box plots or violin plots to analyze the distribution of AQI values.
- **To Relationship Exploration:** Use scatter plots or bubble charts to explore relationships between AQI values and pollutant levels.
- **To Customization:** Add labels, titles, legends, and color schemes to enhance visualizations.

Software and Hardware Requirements:

- **Software:**
  - Python 3.x
  - Libraries: matplotlib, pandas, seaborn (install using pip)
  - Jupyter Notebook (optional, for an interactive environment)
- **Hardware:**
  - A computer with at least 4 GB of RAM
  - Enough storage to handle data files and generate visualizations

Expected Outcome:

1. **Visualized AQI Trends:** Line plots showing overall AQI trends over time.
2. **Pollutant Trends:** Individual plots for PM2.5, PM10, and CO levels.
3. **AQI Comparisons:** Bar plots comparing AQI values across different dates or periods.
4. **Distribution Analysis:** Box plots or violin plots for AQI distribution.
5. **Relationship Insights:** Scatter or bubble charts showing relationships between AQI values andpollutant levels.
6. **Custom Visualizations:** Enhanced visualizations with appropriate labels, titles, and color schemes.

Theory:

Data Visualization involves creating graphical representations of data to help analyze and interpret information. Effective visualization allows for the identification of patterns, trends, and insights that might not be apparent from raw data alone.

1. Line Plots

- **Overview**: Line plots are used to show trends over time or continuous data. They are useful for visualizing how a variable changes across a sequence of intervals.
- **Use Cases**: Tracking stock prices, monitoring temperature changes over days, analyzing sales trends over months.
- **Example**:

```
import matplotlib.pyplot as plt

plt.plot(df['Date'], df['Value'])
plt.xlabel('Date')
plt.ylabel('Value')
plt.title('Trend over Time')
plt.grid(True)
plt.show()
```

2. Bar Plots

- **Overview**: Bar plots are used to compare quantities across different categories or time periods. Theycan be vertical or horizontal.
- **Use Cases**: Comparing sales figures by product category, displaying monthly revenue, analyzing survey responses.
- **Example**:

```
plt.bar(df['Category'], df['Sales'])
plt.xlabel('Category')
plt.ylabel('Sales')
plt.title('Sales by Category')
plt.xticks(rotation=45)
plt.show()
```

3. Box Plots / Violin Plots

- **Overview**:
  - **Box Plots**: Show the distribution of data through quartiles and identify outliers. They areeffective for understanding data spread and central tendency.
  - **Violin Plots**: Combine box plots with kernel density plots to show the distribution of the data across different categories.
- **Use Cases**: Visualizing the spread of test scores, comparing income distribution across different regions.
- **Example**:

```
import seaborn as sns
sns.boxplot(x='Category', y='Value', data=df)
plt.title('Box Plot of Values by Category')
plt.show()
sns.violinplot(x='Category', y='Value', data=df)
plt.title('Violin Plot of Values by Category')
plt.show()
```

4. Scatter Plots / Bubble Charts

- **Overview**:
  - **Scatter Plots**: Display relationships between two variables. They are used to identifycorrelations or patterns.
  - **Bubble Charts**: Extend scatter plots by adding a third dimension with bubble size, representing an additional variable.
- **Use Cases**: Analyzing the relationship between advertising spend and sales, visualizing customer demographics.
- **Example**:

```
plt.scatter(df['Advertising Spend'], df['Sales'])
plt.xlabel('Advertising Spend')
plt.ylabel('Sales')
plt.title('Advertising Spend vs. Sales')
plt.show()
plt.scatter(df['Advertising Spend'], df['Sales'], s=df['Market Share']*100, alpha=0.5)
plt.xlabel('Advertising Spend')
plt.ylabel('Sales')
plt.title('Bubble Chart of Advertising Spend vs. Sales')
plt.show()
```

5. Customization

- **Overview**: Enhancing visualizations to improve readability and interpretation.
- **Techniques**:
  - **Labels**: Adding axis labels, data labels, and annotations to provide context.
  - **Titles**: Including clear and descriptive titles to explain what the plot represents.
  - **Legends**: Adding legends to distinguish between different data series or categories.
  - **Color Schemes**: Applying color schemes to improve visual appeal and clarity.

```
plt.plot(df['Date'], df['Value'], color='blue', marker='o', linestyle='-')
plt.xlabel('Date', fontsize=14)
plt.ylabel('Value', fontsize=14)
```

```
plt.title('Trend over Time', fontsize=16)
plt.legend(['Value'], loc='upper left')
plt.grid(True)
plt.show()
```

Algorithms

1. **Line Plot Trend Analysis**
   - **Moving Average**: Smooth the line plot to highlight trends.

     ```
     df['Moving_Avg'] = df['Value'].rolling(window=7).mean()
     plt.plot(df['Date'], df['Moving_Avg'], label='7-Day Moving Average')
     plt.legend()
     ```

2. **Bar Plot Comparison**
   - **Percentage Change**: Compute and visualize the percentage change between categories.

     ```
     df['Percentage_Change'] = df['Current_Value'] / df['Previous_Value'] * 100
     plt.bar(df['Category'], df['Percentage_Change'])
     ```

3. **Box Plot Outlier Detection**
   - **Identify Outliers**: Use box plots to detect outliers.

     ```
     sns.boxplot(x='Category', y='Value', data=df)
     outliers = df[(df['Value'] > df['Q3'] + 1.5 * (df['Q3'] - df['Q1'])) | (df['Value'] < df['Q1'] - 1.5 * (df['Q3'] - df['Q1']))]
     ```

4. **Scatter Plot Correlation**
   - **Correlation Coefficient**: Calculate the correlation between two variables.

     ```
     correlation = df['Advertising Spend'].corr(df['Sales'])
     plt.scatter(df['Advertising Spend'], df['Sales'])
     plt.title(f'Correlation: {correlation:.2f}')
     ```

5. **Bubble Chart Dimension**
   - **Size Scaling**: Adjust bubble sizes based on an additional variable.

     ```
     plt.scatter(df['Advertising Spend'], df['Sales'], s=df['Market Share']*100, alpha=0.5, c=df['Region'])
     ```

This extended theory on data visualization provides a detailed look at different types of plots, customization techniques, and associated algorithms, along with a flow diagram to guide the visualization process.

Conclusion:

The practical successfully demonstrated the use of Matplotlib to visualize air quality trends and patterns. By creating various types of plots, we were able to analyze AQI trends, observe pollutant levels over time, and explore relationships between different air quality metrics. Customizing the visualizations with appropriate labels, titles, and color schemes enhanced the clarity and interpretability of the results. This approach provides valuable insights into air quality dynamics and supports data-driven decision-making for environmental health.

Viva Questions:

1. What is the Air Quality Index (AQI)?
2. Why is it important to visualize AQI trends?
3. What is the impact of high AQI levels on public health?

Coding Efficiency	Viva	Timely Completion	Total	Dated Sign of Course In-charge
5	3	2	10	

Program:

Experiment No: 6

Data Aggregation and Visualization of Sales Performance by Region

Name of the Student: _____

Class: BE **Batch:**

Date:	Mark: /10

Signature of the Course In-charge: _____

Signature of the HOD: _____

EXPERIMENT NO.6

Practical Title: Data Aggregation and Visualization of Sales Performance by Region

Aim:

To perform data aggregation on sales transactions to analyze and visualize sales performanceby region, and to identify top-performing regions and product categories.

Objectives:

- **To Import Dataset:** Load the "Retail_Sales_Data.csv" dataset.
- **To Explore Data:** Understand the dataset's structure and content.
- **To Identify Variables:** Determine relevant variables for aggregating sales data such as region, sales amount, and product category.
- **To Aggregate Sales Data:** Group the data by region and calculate total sales amounts for each region.
- **To Visualize Sales Distribution:** Create visualizations to represent sales distribution by region.
- **To Top-Performing Regions:** Identify regions with the highest sales amounts.
- **To Sales by Region and Product Category:** Group data by region and product category to analyze total sales amounts.
- **To Compare Sales Performance:** Create visualizations to compare sales across different regions and product categories.

Software and Hardware Requirements:

- **Software:**
  - Python 3.x
  - Libraries: pandas, matplotlib, seaborn (install using pip)
  - Jupyter Notebook (optional, for interactive analysis)
- **Hardware:**
  - A computer with at least 4 GB of RAM
  - Enough storage to handle data files and generate visualizations

Expected Outcome:

1. **Aggregated Sales Data:** Total sales amount per region.
2. **Sales Distribution Visualizations:** Bar plots or pie charts showing sales distribution by region.
3. **Top-Performing Regions:** Identification of regions with the highest sales.

4. **Sales Analysis by Product Category:** Aggregated sales data by region and product category.
5. **Comparison Visualizations:** Stacked bar plots or grouped bar plots comparing sales amounts across regions and product categories.

Theory:

Data Aggregation: Data Aggregation is the process of summarizing and grouping data to derive meaningful metrics and insights. It involves organizing data into groups, calculating key metrics, and visualizing the results to interpret aggregated information effectively.

1. Grouping Data

- **Overview**: Organizing data into groups based on one or more variables. This allows for aggregation and comparison within and across these groups.
- **Use Cases**: Grouping sales data by region, analyzing customer behavior by age group, or summarizing performance metrics by department.
- **Steps**:
  - **Define Grouping Variables**: Determine which columns or variables to use for grouping (e.g., region, product category).
  - **Apply Grouping**: Use functions to group data accordingly.

```
import pandas as pd

# Example dataframe
df = pd.DataFrame({
    'Region': ['North', 'South', 'North', 'East', 'South'],
    'Sales': [200, 150, 300, 250, 100]
})

# Group by 'Region'
grouped_df = df.groupby('Region').sum()
```

2. Calculating Metrics

- **Overview**: Summarizing data within each group to derive meaningful metrics such as totals, averages, and percentages.
- **Common Metrics**:
  - **Total**: Sum of values within each group (e.g., total sales by region).
  - **Average**: Mean of values within each group (e.g., average sales per transaction).
  - **Count**: Number of observations within each group (e.g., number of transactions).
- **Steps**:
  - **Apply Aggregation Functions**: Use functions like `sum()`, `mean()`, `count()`, `median()`, etc.

```
# Calculate total sales by region
total_sales = grouped_df['Sales']
```

```
# Calculate average sales by region
average_sales = df.groupby('Region')['Sales'].mean()
```

3. Visualization

- **Overview**: Creating graphical representations to analyze and interpret aggregated data. Visualizations make it easier to spot patterns, trends, and comparisons.
- **Common Visualizations**:
  - **Bar Plots**: Compare aggregated values across categories.
  - **Pie Charts**: Show proportions of a whole (e.g., market share by region).
  - **Line Charts**: Display trends over time if data is aggregated over time periods.
- **Steps**:
  - **Select Visualization Type**: Choose based on the nature of the data and the insights you want to derive.
  - **Create Visualization**: Use libraries like `matplotlib`, `seaborn`, or `plotly` to generate plots.

```
import matplotlib.pyplot as plt

# Bar plot for total sales by region
plt.bar(total_sales.index, total_sales.values)
plt.xlabel('Region')
plt.ylabel('Total Sales')
plt.title('Total Sales by Region')
plt.show()

# Pie chart for sales distribution by region
plt.pie(total_sales, labels=total_sales.index, autopct='%1.1f%%')
plt.title('Sales Distribution by Region')
plt.show()
```

Algorithms:-

1. **Grouping Data**
   - **Group By**: Aggregate data based on specified criteria.

```
# Grouping by 'Region' and summing 'Sales' grouped_df
= df.groupby('Region')['Sales'].sum()
```

2. **Calculating Metrics**
   - **Total Calculation**:

```
total_sales = df.groupby('Region')['Sales'].sum()
```

   - **Average Calculation**:

```
average_sales = df.groupby('Region')['Sales'].mean()
```

   - **Count Calculation**:

```
count_sales = df.groupby('Region')['Sales'].count()
```

3. **Visualization**
   - **Bar Plot**:

```
plt.bar(grouped_df.index, grouped_df.values)
plt.xlabel('Region')
plt.ylabel('Total           Sales')
plt.title('Total Sales by Region')
plt.show()
```

   - **Pie Chart**:

```
plt.pie(grouped_df, labels=grouped_df.index, autopct='%1.1f%%')
plt.title('Sales Distribution by Region')
plt.show()
```

This extended theory on data aggregation provides a comprehensive look at the process, including grouping data, calculating metrics, visualizing results, and a flow diagram with corresponding algorithms to guide the aggregation and analysis.

Conclusion:

The practical effectively demonstrated data aggregation and visualization techniques to analyze sales performance by region. By grouping sales data and calculating total sales amounts, we identified top-performing regions and compared sales across different product categories. The visualizations, including bar plots, pie charts, and stacked bar plots, provided a clear representation of sales distribution and performance, facilitating better insights into the retail company's sales dynamics. This approach helps in understanding regional sales performance and making data-driven decisions for strategic planning.

Viva Questions:

1. What is data aggregation?
2. Why is data aggregation important in sales performance analysis?
3. Why is data aggregation important in sales performance analysis?

Coding Efficiency	Viva	Timely Completion	Total	Dated Sign of Course In-charge
5	3	2	10	

Program:

Experiment No: 7

Time Series Data Analysis

Name of the Student: _____

Class: BE **Batch:**

Date:

Mark: /10

Signature of the Course In-charge: _____

Signature of the HOD: _____

EXPERIMENT NO.7

Practical Title: Time Series Analysis and Forecasting of Stock Market Data

Aim:

To analyze and visualize historical stock price data to identify trends, patterns, and potential predictors, and to build models to forecast future stock prices.

Objectives:

- **To Import Dataset:** Load the "Stock_Prices.csv" dataset.
- **To Explore Data:** Understand the dataset's structure, content, and ensure date formatting.
- **To Format Date:** Convert the date column to the appropriate datetime format for time series analysis.
- **To Visualize Trends:** Plot line charts or time series plots to show historical stock price trends.
- **To Calculate Moving Averages:** Compute and plot moving averages to identify trends and smooth out noise.
- **To Seasonality Analysis:** Identify and visualize periodic patterns in stock prices.
- **To Correlation Analysis:** Analyze and plot the relationship between stock prices and other variables.
- **To Forecasting Models:** Use ARIMA or exponential smoothing models to forecast future stock prices.

Software and Hardware Requirements:

- **Software:**
  - Python 3.x
  - Libraries: pandas, matplotlib, seaborn, statsmodels (install using pip)
  - Jupyter Notebook (optional, for interactive analysis)
- **Hardware:**
  - A computer with at least 4 GB of RAM
  - Sufficient storage for data files and generated results

Expected Outcome:

1. **Visualized Historical Trends:** Line charts displaying historical stock price trends.
2. **Smoothed Trends:** Plots of moving averages to highlight underlying trends.
3. **Seasonal Patterns:** Visualizations identifying periodic patterns in stock prices.
4. **Correlation Insights:** Analysis and plots showing relationships between stock prices and other variables.
5. **Forecasting Results:** Forecasts of future stock prices using time series models.

Theory:

Time Series Analysis: Time Series Analysis involves examining data points collected or recorded at specific time intervals to uncover underlying patterns, trends, and seasonal effects. This analysis is crucial for forecasting future values and understanding historical data.

1. Trends

- **Overview**: Trends represent the long-term movement in time series data, reflecting the overall direction in which data is moving over an extended period.
- **Identification**: Trends can be upward, downward, or flat. They are often identified through visual inspection of the data or by applying trend-line analysis.
- **Example**: A company's annual revenue growth trend over the past decade.

Diagram:

plaintext
Time Series Data with Trend

```
------------------------------------------------------------
Time |      |      |      |      |      |
Value |---▲---|---▲---|---▲---|---▲---|---▲---|---▲---
      Year 1 Year 2 Year 3 Year 4 Year 5 Year 6
```

2. Moving Averages

- **Overview**: Moving averages smooth out short-term fluctuations in the data to highlight longer-term trends. They are particularly useful for identifying trends and cyclic patterns.
- **Types**:
  - **Simple Moving Average (SMA)**: The average of data points over a fixed period.
  - **Weighted Moving Average (WMA)**: Assigns different weights to data points, giving more importance to recent observations.
  - **Exponential Moving Average (EMA)**: Applies exponentially decreasing weights to past observations, making it more responsive to recent changes.

Example: A 7-day moving average of daily temperatures to identify long-term climate trends.

Diagram:

plaintext
Time Series Data with Moving Average

```
------------------------------------------------------------
Time   |---▲---|---▲---|---▲---|---▲---|---▲---|---▲---|---▲---
       | SMA | SMA | SMA | SMA | SMA | SMA | SMA
Value |    /   |    /   |    /   |    /   |    /   |    /   |    /
       | | | | | | | | | | | | | | |
```

3. Seasonality

- **Overview**: Seasonality refers to periodic fluctuations in time series data that occur at regular intervals due to seasonal factors like time of year, month, or day of the week.
- **Identification**: Seasonality is identified by observing repeating patterns at consistent intervals.

- **Example**: Retail sales often increase during holiday seasons or weekends.

  Diagram:

  plaintext
  Time Series Data with Seasonality

  Time |---▲---|---▼---|---▲---|---▼---|---▲---|---▼---
  | Month 1 | Month 2 | Month 3 | Month 4 | Month 5 | Month 6
  Value | | | | | | | | | | | |

4. Autoregressive Integrated Moving Average (ARIMA)

- **Overview**: ARIMA is a popular forecasting method that combines three components:
  - **Autoregression (AR)**: Uses the dependency between an observation and a number of lagged observations.
  - **Differencing (I)**: Removes trends by differencing the data to make it stationary.
  - **Moving Average (MA)**: Models the dependency between an observation and a residual error from a moving average model applied to lagged observations.
- **Usage**: Suitable for non-stationary time series data and for making forecasts based on past values.

  Diagram:

  plaintext
  ARIMA Model Components

Autoregression (AR) | Residual Error
| .....
| / \
Time Series Data | | Error |
| _____/
| Moving Average (MA)

  Example: Forecasting monthly sales by modeling past sales data using ARIMA.

  Algorithm:

```
from statsmodels.tsa.arima_model import ARIMA

# Fit ARIMA model
model = ARIMA(df['Sales'], order=(p, d, q))
model_fit = model.fit(disp=0)

# Make forecast
forecast = model_fit.forecast(steps=12)
```

5. Exponential Smoothing

- **Overview**: Exponential Smoothing applies decreasing weights to past observations, making recent data more influential for forecasting.
- **Types**:
  - **Simple Exponential Smoothing**: Forecasts based on weighted averages of past observations.
  - **Holt's Linear Trend Model**: Extends simple exponential smoothing to capture linear trends.
  - **Holt-Winters Seasonal Model**: Includes components for trend and seasonality.

Algorithm:

```
from statsmodels.tsa.holtwinters import ExponentialSmoothing#

Fit model
model = ExponentialSmoothing(df['Sales'], seasonal='add', seasonal_periods=12)
model_fit = model.fit()

# Make forecast
forecast = model_fit.forecast(steps=12)
```

Diagram:

plaintext
Copy code

```
Exponential Smoothing
-------------------------------------------------------------
Time |.....|.....|.....|.....|.....|.....|.....|
     | Latest | Past | Past | Past | Past
     | Observations | Observations | Observations
Weight | 0.7 | 0.2 | 0.05 | 0.025 | 0.005
```

Summary

- **Trends** help identify long-term movements.
- **Moving Averages** smooth out fluctuations to reveal trends.
- **Seasonality** captures regular, periodic fluctuations.
- **ARIMA** combines autoregression, differencing, and moving averages for forecasting.
- **Exponential Smoothing** applies weighted averages to forecast future values.

This extended theory provides a comprehensive understanding of time series analysis, complete with diagrams and algorithms for practical application

Conclusion:

The practical demonstrated the application of time series analysis techniques on stock market data. By

visualizing historical trends, calculating moving averages, and analyzing seasonal patterns, we gained insights into the stock price behavior. The correlation analysis revealed relationships between stock prices and trading volume, while ARIMA and Exponential Smoothing models provided forecasts for future stock prices. These techniques and visualizations offer valuable tools for understanding stock price dynamics and making informed investment decisions.

Viva Questions:

1. What is a time series?
2. What distinguishes a time series from cross-sectional data?
3. What is a white noise process in time series analysis?

Coding Efficiency	Viva	Timely Completion	Total	Dated Sign of Course In-charge
5	3	2	10	

Program: